UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JONATAS FERREIRA DE LIMA

TERRASCRATCH: APRENDENDO TERRAFORM DE MANEIRA INTERATIVA

CAMPINA GRANDE - PB

2024

JONATAS FERREIRA DE LIMA

APRENDENDO TERRAFORM DE MANEIRA INTERATIVA: TERRASCRATCH

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Thiago Emmanuel da Silva

CAMPINA GRANDE - PB

2024

JONATAS FERREIRA DE LIMA

TERRASCRATCH: APRENDENDO TERRAFORM DE MANEIRA INTERATIVA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Thiago Emmanuel da Silva

Orientador – UASC/CEEI/UFCG

Reinaldo Cézar de Morais Gomes

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de maio de 2024.

CAMPINA GRANDE - PB

RESUMO

Terraform é uma ferramenta popular que permite o provisionamento de infraestrutura através de código. Pelo fato de ser algo recente, materiais intuitivos e de aprendizado são escassos, se limitando a vídeos tutoriais e documentações tradicionais, o que afasta o usuário iniciante que tem interesse em aprender essa ferramenta. Com base nesse fato, foi desenvolvido o *Terrascratch:* uma ferramenta intuitiva que facilita o processo de aprendizado. O *Terrascratch* permite o uso de recursos visuais que representam componentes de infraestrutura da nuvem da *Amazon Web Services (AWS)* e gera o seu código *Terraform* correspondente. O levantamento de requisitos para a ferramenta foi realizado por meio de um *workshop*, e a eficácia da ferramenta foi avaliada mediante experimentos qualitativos com usuários potenciais em um segundo *workshop*. Realizamos uma apresentação e uma série de entrevistas com os participantes. Como resultados, observamos que o *Terrascratch* é eficaz para usuários sem conhecimento prévio de *Terraform*, com destaque para os usuários com noções básicas de AWS, mas não tão útil para usuários de conhecimento intermediário ou avançado.

TERRASCRATCH: LEARNING INFRASTRUCTURE AS CODE INTERACTIVELY

ABSTRACT

Terraform is a popular tool that allows for infrastructure provisioning through code. As it is a relatively new technology, intuitive and educational materials are scarce, limited to tutorial videos and traditional documentation, which discourages beginners interested in learning this tool. Based on this fact, Terrascratch was developed: an intuitive tool that facilitates the learning process. Terrascratch enables the use of visual elements representing Amazon Web Services (AWS) cloud infrastructure components and generates the corresponding Terraform code. The requirements gathering for the tool was conducted through a workshop, and its effectiveness was assessed via qualitative experiments with potential users in a second workshop. We conducted a presentation and a series of interviews with the participants. As a result, we observed that Terrascratch is effective for users without prior knowledge of Terraform, particularly for those with basic AWS knowledge, but not as useful for users with intermediate or advanced knowledge.

Terrascratch: Aprendendo Terraform de maneira interativa

Autor
Jonatas Ferreira de Lima
Universidade Federal de Campina Grande
jonatas.lima@ccc.ufcg.edu.br

Orientador
Thiago Emmanuel Pereira
Universidade Federal de Campina Grande
temmanuel@computacao.ufcg.edu.br

RESUMO

Terraform é uma ferramenta popular que permite o provisionamento de infraestrutura através de código. Pelo fato de ser algo recente, materiais intuitivos e de aprendizado são escassos, se limitando a vídeos tutoriais e documentações tradicionais, o que afasta o usuário iniciante que tem interesse em aprender essa ferramenta. Com base nesse fato, foi desenvolvido o *Terrascratch:* uma ferramenta intuitiva que facilita o processo de aprendizado. O *Terrascratch* permite o uso de recursos visuais que representam componentes de infraestrutura da nuvem da *Amazon Web Services (AWS)* e gera o seu código *Terraform* correspondente. O levantamento de requisitos para a ferramenta foi realizado por meio de um *workshop*, e a eficácia da ferramenta foi avaliada mediante experimentos qualitativos com usuários potenciais em um segundo *workshop*. Realizamos uma apresentação e uma série de entrevistas com os participantes. Como resultados, observamos que o *Terrascratch* é eficaz para usuários sem conhecimento prévio de *Terraform*, com destaque para os usuários com noções básicas de AWS, mas não tão útil para usuários de conhecimento intermediário ou avançado.

Palavras-chave

Terraform, AWS, *Cloud*, aprendizado, *no-code*, *DevOps*.

Repositório

https://github.com/terrascratch/terrascratch

1. INTRODUÇÃO

Tradicionalmente, empresas dividem seu departamento de tecnologia em duas equipes: desenvolvimento e operações. A equipe de desenvolvimento é responsável por desenvolver e testar o *software* para a empresa, o que garante que o produto entregue seja funcional. Já o time de operações é responsável por implantar esse *software* e garantir que ele esteja funcionando corretamente.

De acordo com Gene Kim, esses dois times desempenham papéis antagônicos [1], com diferentes metas e prioridades, o que leva a falhas na comunicação, falta de alinhamento entre os times e a geração de conflitos, que resulta em atrasos nas entregas do produto e na deterioração da sua qualidade.

Como tentativa de resolver esses problemas, foi criado o *DevOps* [1], uma cultura de colaboração — entre os times de desenvolvimento e de operações — com o propósito de entregar um *software* com a qualidade e rapidez que o mercado exige. Isso é feito através do compartilhamento de responsabilidades e de técnicas como automação de processos, de integração e entrega contínua (CI/CD), do monitoramento e da infraestrutura como código.

Esse último conceito, *Infrastructure as Code* [2] *(IaC),* garante que a infraestrutura virtual — seja na nuvem pública ou *on-premises* — da organização esteja visível para todo o time de tecnologia; e que, caso uma alteração errada seja feita, um simples *rollback* de *commit* vai voltar ao estado que era antes. Em resumo, assegura a agilidade e a reprodutibilidade no provisionamento da infraestrutura da empresa.

Uma das principais ferramentas de *IaC* é o *Terraform* [3], que permite a escrita deste código, de forma declarativa, com arquivos de configuração em linguagem natural — geralmente JSON ou HCL (*Hashicorp Configuration Language*) — de maneira padronizada, rastreável e versionada.

O conceito de *IaC* e o *Terraform* são relativamente recentes — e o conhecimento dessa ferramenta é cada vez mais essencial para um profissional se destacar no mercado de trabalho — o que reflete na escassez de materiais de aprendizado interativos — como o Python Tutor [4] ou o Scratch [5] para linguagens de programação —, se limitando às fontes tradicionais como vídeos tutoriais e documentações, o que torna o aprendizado desses conceitos lento, pouco atrativo e pouco difundido na comunidade.

Além da escassez de material atrativo, há, também, a complexidade na escrita do *HCL* para pessoas iniciantes no tema, já que é necessário ter um conhecimento prévio dos recursos de infraestrutura da provedora de nuvem para a qual será escrito o código.

Como será visto posteriormente, essa hipótese da dificuldade foi levantada através de um *workshop* realizado com pessoas que têm interesse em aprender *Terraform*, mas sem experiência prévia, alunos do curso de Ciência da Computação na Universidade Federal de Campina Grande (UFCG), o que serviu, também, para o levantamento de requisitos do *Terrascratch*.

Em relação às dificuldades de aprendizado do *Terraform* levantadas pelos participantes deste *workshop*, pode-se citar:

- A vasta gama de conhecimentos prévios necessários;
- A falta de conhecimento do *cloud provider*;
- A imensa documentação do *Terraform*.

Com base nessas informações, o *Terrascratch* foi implementado. O objetivo principal foi facilitar o acesso à documentação do *Terraform*, além de breves explicações acerca dos recursos da AWS disponíveis na ferramenta.

Nesse contexto, a proposta da ferramenta é de, através de uma interface gráfica no formato de *playground,* totalmente *no-code,* permitir que o usuário manipule componentes visuais — correspondentes a recursos de infraestrutura na provedora de nuvem *Amazon Web Services* (AWS) — e veja sua representação em *Terraform* (mais especificamente HCL).

Por fim, em um segundo *workshop,* foi observado que essa abordagem facilitou o aprendizado desta ferramenta para as pessoas que pretendem aprender *IaC* com *Terraform,* principalmente as que já possuem conhecimentos básicos de redes e de AWS.

Na seção 2, será feita a introdução dos conceitos de computação em nuvem, além de *Infrastructure as Code* e *Terraform.* Em seguida, na seção 3, a implementação da ferramenta será detalhada. Logo após, na seção 4, serão abordados os métodos de avaliação da ferramenta desenvolvida, que servirão de base para a discussão dos resultados na seção 5. A partir da discussão dos resultados obtidos, na seção 6, teremos as conclusões, além da discussão da possibilidade da realização de trabalhos futuros.

2. CONTEXTUALIZAÇÃO

Nesta seção, serão abordados os conceitos de computação em nuvem de forma detalhada, com a justificativa de escolha do provedor de nuvem AWS. Logo após, alguns recursos de infraestrutura da AWS que estão disponíveis no *Terrascratch* serão detalhados. Por fim, será apresentado o conceito de *Infrastructure as Code*, com a ferramenta *Terraform* e como ela se encaixa no contexto de *IaC*, além de um breve resumo sobre o seu funcionamento.

2.1 CLOUD COMPUTING

Quando se trata de implantação de sistemas, fala-se tipicamente em dois ambientes: *on-premises* e *cloud* pública. Em um ambiente *on-premises*, a empresa é totalmente responsável por toda a infraestrutura e recursos de computação. Isso inclui a compra dos servidores, a aquisição de um local físico para armazená-los, o pagamento de energia elétrica, a contratação de provedores de internet e a manutenção do software que será executado nas máquinas, entre outras responsabilidades. Por outro lado, em um ambiente de *cloud* pública, a empresa utiliza esses recursos computacionais fornecidos por um provedor especializado, acessando-os pela internet e pagando apenas pelo que utiliza. Nesse modelo, a responsabilidade pela manutenção da infraestrutura é do provedor de nuvem, o que simplifica a gestão para a empresa contratante.

A computação em nuvem oferece diversos benefícios, como a redução do tempo gasto em tarefas operacionais, como organizar racks, virtualizar máquinas, realizar cabeamento e ajustar a temperatura do data center. Isso resulta em uma maior rapidez na disponibilização de recursos e um aumento na disponibilidade dos serviços, devido à arquitetura globalmente distribuída da maioria dos provedores de nuvem. Além disso, há a vantagem do pagamento sob demanda.

Em suma, a computação em nuvem é a entrega de serviços de computação, como servidores, armazenamento, bancos de dados, redes, *softwares* através da internet. Esse fornecimento é feito por uma empresa especializada.

Atualmente, uma das maiores fornecedoras de computação em nuvem é a Amazon Web Services (AWS), com 48,62% dos usuários de provedores de nuvem que responderam à pesquisa do Stack Overflow em 2023. Devido à sua ampla utilização no mercado e popularidade, a AWS será utilizada como referência para a geração do código Terraform pelo Terrascratch.

2.2 RECURSOS DE INFRAESTRUTURA NA AWS

A seguir, serão listados — com uma breve explicação — os recursos de infraestrutura fornecidos pela AWS que estarão disponíveis para geração de código no *Terrascratch*. É importante ter um conhecimento prévio — ao menos básico — acerca desses recursos, para usufruir por completo da ferramenta.

2.2.1 VIRTUAL PRIVATE CLOUD (VPC)

Em relação aos serviços de rede, a AWS fornece ao usuário uma seção — logicamente isolada — da sua rede, com agrupamentos (subconjuntos) públicos ou privados, os quais são chamados de sub-redes.

É necessário que, para uma máquina virtual da AWS ser acessível, ela esteja associada a uma VPC — e, consequentemente, em uma sub-rede pública ou privada. É o recurso primordial para a criação de todos os outros recursos.

Quando se trata de uma sub-rede pública, outros componentes são criados automaticamente pelo *Terrascratch*:

2.2.2 SECURITY GROUP (SG)

Em relação ao tráfego de entrada e de saída nas instâncias da AWS, o recurso que irá ditar essas regras é o *Security Group*, um *firewall* virtual fornecido pela AWS. Por padrão, todas as portas, em todas as direções, em todos os protocolos, são bloqueadas.

Os protocolos de comunicação suportados nos *Security Groups* são o *Transmission Control Protocol* (TCP) — utilizado para conexões confiáveis, com retransmissão e tratamento de erros —, o *User Datagram Protocol* (UDP) — utilizado para conexões não-confiáveis — e o *Internet Control Message Protocol* (ICMP) — utilizado para relatório de erros.

Para criar uma regra no SG é necessário:

- **Protocolo de comunicação**;
- **Direção** (entrada ou saída na máquina);
- **Porta de rede**;
- **Endereço IP** (de origem ou de destino).

2.2.3 ELASTIC COMPUTE CLOUD (EC2)

É o serviço de computação virtual. Isto é, representa uma máquina virtual, ou um servidor físico dedicado, gerenciado pela AWS, o qual irá executar o *software* do usuário. A provedora toma de conta dos processos de virtualização e de manutenção para o cliente.

É possível configurar algumas características da instância, como:

- ***Amazon Machine Image* (AMI)**: Sistema Operacional da instância;
- **Tipo**: Quantidade de CPU e de Memória RAM;

- ***Security Groups***: A quais *security groups* a instância pertence;
- **Sub-rede**: Em qual sub-rede a instância está situada.

2.3 INFRASTRUCTURE AS CODE (IAC)

O processo de provisionamento da infraestrutura virtual da empresa pode ser feito de algumas formas, dentre elas: o time de operações cria os recursos de forma ad-hoc na interface gráfica da provedora — conhecido na comunidade informalmente como *ClickOps* — ; ou o time de operações desenvolve *scripts* customizados que realizam esses procedimentos de forma automática — geralmente de forma imperativa.

A primeira forma de provisionamento é bastante propensa a erros humanos, visto que é bem provável que, ao se tentar reproduzir, caso não haja uma documentação clara, o usuário esqueça alguma etapa, ou preencha alguma configuração errada, sendo uma alternativa sem rastreabilidade, manual e pouco escalável. Já a segunda forma, o provisionamento é feito de maneira automática, possui rastreabilidade — visto que os *scripts*, muito provavelmente, estarão em um repositório de gerenciamento de código, como o Github — mas continuam propensos a erros, já que uma abordagem imperativa não garante que, ao ser executado duas ou mais vezes, o *script* vai criar os mesmos recursos — conceito conhecido como idempotência.

Para resolver os problemas das duas abordagens, o time de operações define a infraestrutura virtual em arquivos de configuração, padronizados, escritos de forma declarativa e em linguagem natural.

Essa última alternativa garante a previsibilidade no provisionamento do ambiente — por estar escrito de forma declarativa — e facilita a distribuição e edição das configurações além de, também tornar a infraestrutura virtual da empresa em um projeto de software como qualquer outro, garantindo rastreabilidade, auditoria e a aplicação dos processos tradicionais da engenharia de software. Esse é o conceito de *Infrastructure as Code* (IaC).

2.4 TERRAFORM

A ferramenta mais utilizada no mercado atualmente para *IaC* é o *Terraform*. Criado pela empresa *Hashicorp* em 2014, a ferramenta lê arquivos de configuração — geralmente escritos em *HCL* (*Hashicorp Configuration Language*) —, traduz para chamadas às *APIs* dos provedores de nuvem escolhidos, e cria, remove ou atualiza os recursos definidos.

A linguagem *HCL*, na qual os recursos são definidos, foi desenvolvida com o objetivo de abordar de forma declarativa o estado da infraestrutura da empresa e de diferenciar o código de infraestrutura do código de aplicação.

```
resource "aws_vpc" "main" {
  cidr_block = var.base_cidr_block
}

<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

Figura 1. *Estrutura básica de um bloco de recurso em HCL.*

Em resumo, o *Terraform* lê o estado definido nos arquivos *HCL*, se comunica com a provedora de nuvem, calcula as diferenças entre o estado dos arquivos e o estado da infraestrutura e realiza as operações necessárias para que esses estados estejam em concordância.

3. DESIGN E IMPLEMENTAÇÃO

Nesta seção, as principais funcionalidades da aplicação do *Terrascratch* serão abordadas, junto com os detalhes de implementação.

3.1 PRINCIPAIS FUNCIONALIDADES

A principal funcionalidade da ferramenta é fornecer uma representação visual dos componentes de infraestrutura da AWS que foram descritos anteriormente.

O *Terrascratch* divide a tela em duas: do lado esquerdo o usuário manipula os recursos visuais, e do lado direito o código *Terraform* correspondente é renderizado em tempo real. Isso permite que o usuário associe o que está sendo criado de forma visual, do lado esquerdo, ao código gerado, do lado direito.

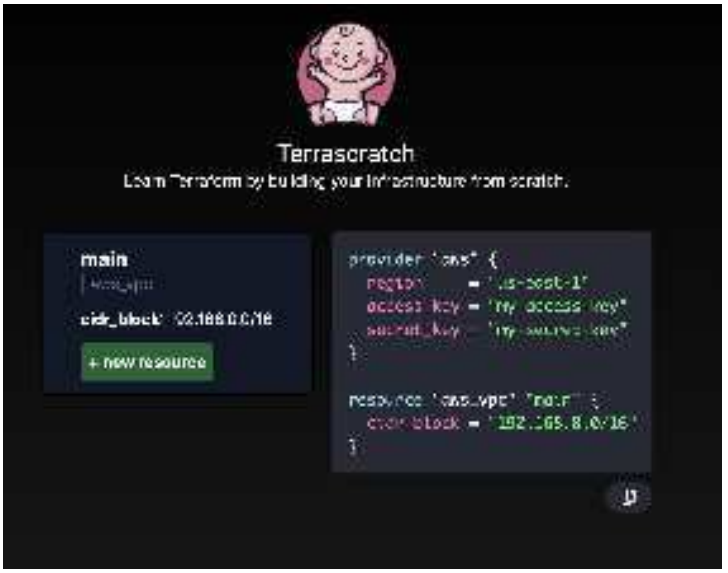

Figura 2. *Interface primária*

Cada recurso tem a possibilidade de adicionar um recurso "filho" ou de deletar a si próprio, juntamente com seus "filhos". Como a estrutura de dados é uma árvore, ao remover um recurso, é removida toda a sub-árvore em que a raiz é o nó a ser deletado.

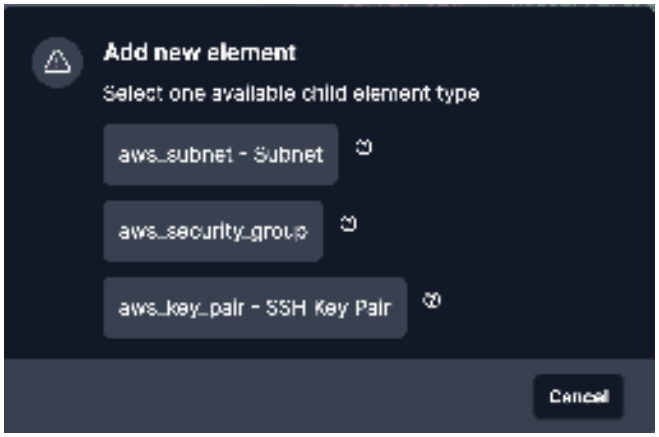

***Figura 3**. Recursos-filho disponíveis a uma VPC*

***Figura 4**. Código Terraform refletindo os recursos à esquerda*

Para alguns recursos, há a possibilidade de selecionar um *template* pré-definido, que preenche automaticamente os valores das propriedades do recursos. Como na seção de *Security Group,* é possível escolher as seguintes pré-configurações:

***Figura 5**. Pré-configurações para uma regra do Security Group*

Por fim, cada recurso, junto com seus atributos, possuem uma breve documentação, apresentada ao usuário ao passar o cursor no ícone de ajuda.

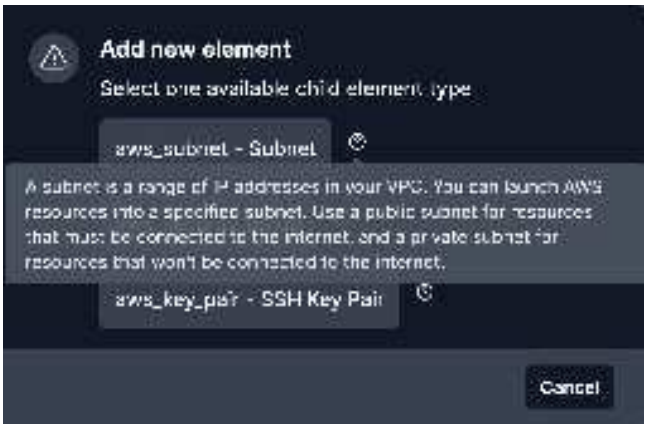

***Figura 6**. Documentação do recurso aws_subnet ao passar o cursor no ícone de ajuda.*

Após o código ser renderizado, o usuário tem a possibilidade de copiá-lo e usá-lo como bem entender.

3.2 DETALHES DE IMPLEMENTAÇÃO

Para facilitar o uso e o acesso do *Terrascratch* aos usuários, optou-se por desenvolver uma aplicação *web*, totalmente *front-end,* que executa por completo no navegador do usuário, com *Next.js [7]* e *TypeScript [8],* devido ao conhecimento prévio dessas tecnologias, o que permitiu uma implementação rápida e sem muitas dificuldades, além de serem tecnologias já estabelecidas no mercado, o que facilita, também, novas contribuições de outras pessoas.

A estrutura de dados utilizada para representar o código *Terraform* foi uma árvore, em que cada nó da estrutura representa um recurso de infraestrutura, o qual está definido num arquivo de configuração no formato JSON, contendo as seguintes informações:

- Nome do recurso;
- Documentação do recurso;
- Atributos;
  - Nome do atributo;
  - Tipo do atributo;
  - Documentação do atributo;
  - Configuração de *input*;
- Outros recursos "filhos".

Com a árvore construída, um *parser* lê a estrutura de dados e transpila em código *Terraform,* renderizando em tempo real na tela do usuário.

4. MÉTODO

Nesta seção, a metodologia para levantamento de requisitos e para a avaliação da ferramenta desenvolvida serão detalhadas, com a descrição dos participantes de ambas as etapas.

Para o levantamento de requisitos, foi realizado um *workshop* em outubro de 2023, com aproximadamente 50 participantes — todos alunos da graduação de Ciência da Computação da Universidade Federal de Campina Grande (UFCG) — e, ao final, foram levantadas as principais dificuldades na hora de desenvolver algum código *Terraform*. O *workshop* foi um dos sub-eventos do *Hacktoberfest* UFCG, um evento anual organizado pela *OpenDevUFCG* que promove palestras, *workshops* e minicursos de tecnologia para o curso de Ciência da Computação da UFCG.

Em abril de 2024, com a ferramenta já minimamente desenvolvida, foi realizado um segundo *workshop* com a apresentação do *Terrascratch*, dessa vez para um público-alvo específico selecionado — 12 estudantes do 3º ao 9º período da graduação de Ciência da Computação na UFCG. Estes participantes foram convidados devido ao prévio interesse demonstrado no primeiro evento. Neste *workshop*, foi realizada uma atividade que consistia em criar uma infraestrutura básica na AWS com a ferramenta, sem um guia de como fazer, apenas com a descrição da infraestrutura a ser criada.

Além dos 12 estudantes, também foi incluído na pesquisa um engenheiro de software já inserido no mercado do trabalho e que já possui certo domínio em relação ao *Terraform* e AWS.

Durante a atividade, foram realizadas anotações — de maneira não-estruturada — acerca do processo de criação da infraestrutura por parte dos estudantes.

Para coletar feedbacks sobre a ferramenta e avaliar os conhecimentos dos participantes, foi elaborado um Google Forms, o qual foi devidamente respondido pelos participantes do workshop e por mais uma pessoa já inserida no mercado de trabalho. Foi utilizada a Escala de *Likert* para avaliação das perguntas quantitativas.

5. DISCUSSÃO

Nesta seção, os resultados obtidos a partir da pesquisa de avaliação com os participantes do segundo *workshop* serão discutidos.

No que tange aos conhecimentos abordados, no quesito redes, dos 13 participantes, cinco se classificaram com conhecimentos básicos, cinco com conhecimentos intermediários e três com conhecimentos avançados.

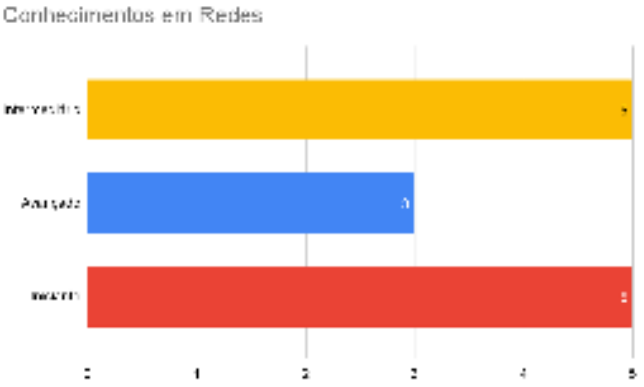

***Figura 7**. Grau de conhecimento em redes dos participantes.*

Em relação à AWS, oito se classificaram com conhecimentos básicos, quatro com conhecimentos intermediários e apenas um com conhecimentos avançados. Todos os participantes já tinham conhecimento — em algum grau — com AWS.

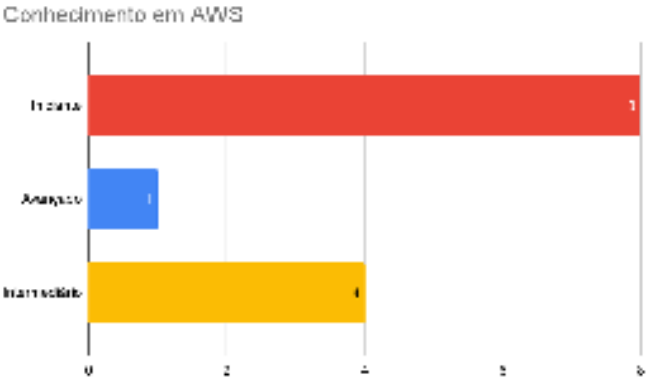

***Figura 8**. Grau de conhecimento em AWS.*

Por fim, em relação ao *Terraform*, pouco mais de dois terços nunca tiveram contato, dois se classificaram iniciantes e dois intermediários.

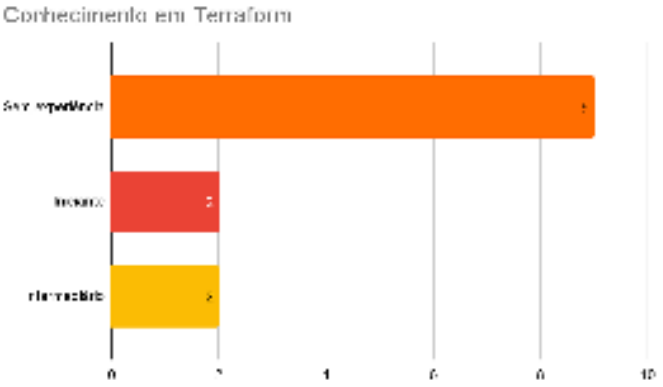

***Figura 9**. Grau de conhecimento em Terraform dos participantes.*

Observou-se que, dentre os participantes sem conhecimento prévio de *Terraform*, os que não tinham conhecimento de AWS tiveram uma maior dificuldade — como esperado — em relação aos outros participantes.

Tal dificuldade advém, principalmente, da falta de conhecimento do *cloud provider*, visto que os participantes sem conhecimento prévio de *Terraform*, mas com conhecimentos de AWS, conseguiram performar a atividade sem maiores dificuldades.

No entanto, em teoria, a documentação da ferramenta deveria facilitar seu uso, mesmo para usuários sem experiência com o *cloud provider*, o que sugere uma falta de intuitividade em relação às ajudas presentes na ferramenta.

Dentre os participantes que já tinham conhecimento sobre o *Terraform*, a atividade foi feita sem dificuldades, também como esperado. Todos acharam a ferramenta bastante intuitiva.

Ao serem questionados sobre se a ferramenta ajudou — ou ajudaria — no seu aprendizado de *Terraform*, todos que participaram do workshop responderam que sim, com a maioria dos feedbacks positivos, tendo como feedbacks construtivos melhorias na interface gráfica e a adição de um botão de editar. Algumas respostas relevantes podem ser vistas no Apêndice 2.

Já em relação aos participantes mais experientes com *Terraform*, a ferramenta não pareceu muito útil, pois o código gerado é muito básico — isto é, só gera infraestruturas básicas —, mas que com o incremento de novos recursos, como volumes (EBS), load balancers (ELB), entre outros, a possibilidade de uso seria maior.

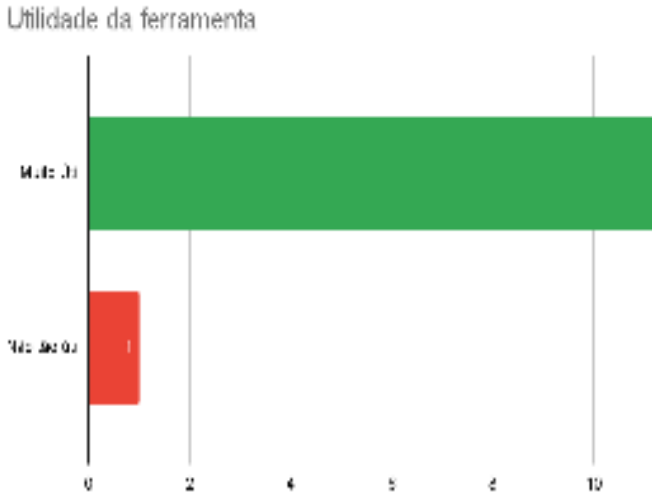

***Figura 10**. Utilidade da ferramenta para o aprendizado de Terraform.*

O único *feedback* não-positivo da ferramenta veio do engenheiro de software já inserido no mercado.

6. CONCLUSÕES

Nesta seção, as conclusões obtidas a partir dos resultados previamente discutidos serão dissertadas de forma detalhada. Ao final da seção, será abordada a possibilidade de trabalhos futuros para extensão e melhoria do *Terrascratch*.

O *Terrascratch* cria um código simples, com alguns dos principais componentes da AWS, como máquinas virtuais (EC2), redes e sub-redes (VPC), firewalls (*Security Group*), roteadores (IGW) e tabelas de roteamento. Por esse motivo, ela funciona como ferramenta de aprendizado, não sendo indicada para geração de código *Terraform* para produção, nem para pessoas que já têm experiência nessa ferramenta.

Acerca do público-alvo, o *Terrascratch* se mostrou mais eficaz nas pessoas que já possuem um entendimento básico dos componentes da AWS, os quais foram mencionados logo acima. Mesmo com usuários que nunca haviam tido contato com *Terraform* — mas que tinham experiência com AWS — conseguiram usufruir do máximo da ferramenta e entender o código gerado.

As dificuldades encontradas com os participantes que não possuíam nem conhecimento em *Terraform*, nem em AWS, sugerem que a forma com que as documentações estão dispostas na ferramenta não esteja facilitando a criação do código *Terraform*. Então, faz-se necessária a busca por maneiras mais intuitivas e mais chamativas do usuário consultar a documentação na ferramenta.

Em relação aos trabalhos futuros, para incrementar a ferramenta, há diversas possibilidades, as quais se concentram em duas principais:

- Adicionar novos componentes;
- Adicionar novos *cloud* providers.

A curto prazo, é viável adicionar novos recursos de infraestrutura da AWS na ferramenta, como:

- **Volumes** (*Elastic Block Storage* - EBS);
- ***Load Balancers*** (*Elastic Load Balancer* - ELB);
- ***Buckets*** (*Simple Storage Service* - S3).

A longo prazo, é interessante a adição de alguns *cloud* providers, como:

- ***OpenStack*** (Utilizado na UFCG, abrindo a possibilidade de o Terrascratch ser usado em alguma disciplina da graduação);
- ***Google Cloud Provider*** (GCP).

Em suma, a ferramenta não é indicada para gerar código Terraform de produção; não vai surtir efeito em pessoas que já têm uma experiência considerável com *Terraform*; é indicada — principalmente — para as pessoas que têm noção de AWS e querem aprender *Terraform*; possui um potencial de expansão considerável — e está aberto às contribuições da comunidade — e, por fim, é indicado para pessoas que não têm experiência com AWS, mas que estejam cientes que o recomendado é ter noções básicas do *cloud provider*.

7. AGRADECIMENTOS

Em primeiro lugar — seguindo os clichês — gostaria de agradecer a mim mesmo, por ter perseverado diante das dificuldades da vida e por ter me dado a oportunidade de cursar uma graduação que

acabei me encontrando muito, e que tenho certeza que foi uma das escolhas mais certas da minha vida. Em segundo lugar, agradeço aos meus pais, Quidenilta e Valeriano — a.k.a Kildinha e Valério —, que são minha base, hoje e sempre, que me ajudam em todos os momentos, em todos os aspectos da vida; sem eles nada seria possível. Agradeço ao meu orientador, Thiago Emmanuel, por, pacientemente, me guiar nesses últimos passos da graduação. Por fim, mas não menos importante, gostaria de agradecer às amizades que transformaram essa caminhada numa aventura, com recordações que se cravaram eternas no meu hipocampo. Hiarly, Luiz, Kleber, Vitinho, Bernard, Davi, Paola, Samuel e Marcos, obrigado por todas as manhãs, tardes e noites no 1301. Davi, obrigado por me auxiliar no desenvolvimento da ferramenta, você é o cara. Pedro, Raiani e Maely, obrigado por todas as conversas profundas do absoluto nada. Maria, obrigado por me ajudar a escrever esse documento e pelo seu apoio incondicional em tudo na minha vida. Obrigado a todos pelo seu amor, nunca vou esquecer de vocês. Em memória de Adriano, eterno em nossos corações.

8. REFERÊNCIAS

[1] KIM, Gene et al. **Manual de DevOps: como obter agilidade, confiabilidade e segurança em organizações tecnológicas**. Rio de Janeiro, Rj: Alta Books, 2018. 427 p.

[2] RED HAT (org.). **What is Infrastructure as Code (IaC)?** 2022. Disponível em: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac. Acesso em: 09 abr. 2024.

[3] HASHICORP (org.). **What is *Terraform*?** Disponível em: https://developer.hashicorp.com/terraform/intro. Acesso em: 09 abr. 2024.

[4] PYTHON TUTOR (org.). **Online Compiler, Visual Debugger, and AI Tutor for Python, Java, C, C++, and JavaScript**. Disponível em: https://pythontutor.com/. Acesso em: 09 abr. 2024.

[5] SCRATCH (org.). **Scratch - Imagine, Program, Share**. Disponível em: https://scratch.mit.edu/. Acesso em: 09 abr. 2024.

[6] STACKOVERFLOW (org.). **2023 Developer Survey**. 2023. Disponível em: https://survey.stackoverflow.co/2023/#overview. Acesso em: 09 abr. 2024.

[7] NEXTJS (org.). **The React Framework for the Web**. Disponível em: https://nextjs.org/. Acesso em: 15 abr. 2024.

[8] TYPESCRIPT. **What is TypeScript?** Disponível em: https://www.typescriptlang.org/. Acesso em: 15 abr. 2024.

9. APÊNDICES

Nesta seção, estão as perguntas e respostas do formulário utilizado para coletar os dados necessários para a avaliação do *Terrascratch.*

APÊNDICE 1 - Perguntas do formulário de avaliação.

1.1. Como você classifica seu conhecimento em redes?

1.2. Como você classifica seu conhecimento em AWS?

1.3. Como você classifica seu conhecimento em Terraform?

1.4. Na sua opinião, quais as principais dificuldades na hora de implementar algo em Terraform/HCL?

1.5. O quão intuitiva a ferramenta é?

1.6. O quão útil o Terrascratch foi no aprendizado de Terraform?

1.7. De acordo com as dificuldades que você apontou, como o Terrascratch ajudou a saná-las?

1.8. Você usaria o Terrascratch para auxiliar no aprendizado de Terraform? Ou para implementar pequenas infraestruturas?

1.9. Feedback? Alguma sugestão de melhoria? Espaço livre para falar o que quiser.

APÊNDICE 2 - Respostas relevantes do formulário de avaliação.

2.1. Achei a ferramenta muito incrível e ela facilita bastante o primeiro contato com Terraform. A sugestão seria continuar incrementando a ferramenta com outros componentes presentes na AWS.

2.2. A ferramenta é extremamente fácil de ser usada, realmente gostei bastante. Se o autor for continuar com essa ferramenta, poderia ser inserido uma opção para editar algum módulo já criado (atualmente temos que deletar e em seguida criar o módulo novamente).

2.3. A ferramenta é uma ótima ideia e muito útil, sem dúvidas. Acredito que ela pode ser mais útil para quem quer implementar uma infraestrutura de menor escala e apenas dar um copy paste no código gerado. Seria interessante se de alguma forma fosse instigado no usuário que ele poderia revisar o código gerar e identificar como ele faria uma declaração de VPC ou de uma instância, algo mais geral.

2.3. Um feedback quanto à usabilidade, um botão para edição seria muito interessante, mas a ferramenta é intuitiva. Uma ótima adição seria mais templates, acho que facilita muito na hora de montar. No mais, muito bom, parabéns!

2.4. Vale usar pra algo básico. Para algo mais avançado faltam certas features